**ОБГРУНТУВАННЯ ВИБРАНОГО НАПРЯМКУ**

**Алгоритм Вітербі**

Алгоритм Вітербі реалізує оптимальне (максимально правдоподібне) декодування як рекурентний пошук на кодової решітці шляху, найближчого до прийнятої послідовності. На кожній ітерації алгоритму Вітербі зіставляються два шляхи, що ведуть в даний стан (вузол решітки). Найближчий з них до прийнятої послідовності зберігається для подальшого аналізу як «виживший», тоді як інший відкидається. Таким чином, якщо ігнорувати випадки, коли обидва конкуруючих шляхи рівновіддалені від прийнятої послідовності, то число «виживших» шляхів, які зберігаються в пам'яті, дорівнює числу вузлів 2*т*. Основні операції алгоритму пояснюються для коду з прикладу 1.

Нехай передається нульове кодове слово, а в каналі сталася триразова помилка, так що прийнята послідовність має вигляд 10 10 00 00 10 00 ... 00 .... Результати пошуку найближчого шляху після прийому 14 елементарних блоків показані на рис. 4.

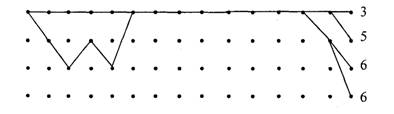


Рис. 4. Приклад роботи алгоритму Вітербі

На правій частині малюнка видно чотири шляхи, що ведуть в кожен вузол решітки. Поруч проставлені метрики (Хеммінгові відстані цих шляхів від прийнятої послідовності на відрізку з 14 блоків). Метрика верхнього шляху значно менше метрик нижніх. Тому можна припустити, що верхній шлях найбільш імовірний. Однак декодер Вітербі, не знаючи наступних фрагментів прийнятої послідовності, змушений запам'ятати всі чотири шляхи на час прийому L елементарних блоків. Число L називається шириною вікна декодування. Для зменшення помилки декодування величину L слід вибирати досить великою, в багато разів перевищуючу довжину кодового обмеження, що природно ускладнює декодер. В даному випадку L = 15.

Відзначимо, що тактика вибору і подальшого аналізу тільки одного шляху з найменшою відстанню складає сутність більш економного послідовного декодування.

На середній частині рис. 4 видно, що всі шляхи мають загальний відрізок (зливаються від 5-го до 12-го кроку) і, отже, прийом нових блоків не може вплинути на конфігурацію цієї ділянки найбільш правдоподібного шляху. Тому декодер вже може приймати рішення про значення інформаційних символів, відповідних цим елементарним блокам.

Ліва частина малюнка демонструє можливу ситуацію непоправної помилки. Існує два шляхи з однаковими метриками. Декодер може розв'язати цю невизначеність двома способами: відзначити цю ділянку як недостовірну або прийняти одне з двох конкуруючих рішень (інформаційна послідовність дорівнює 00000 ... або 10 100 ...). Очевидно, що розширення вікна декодування не дозволяє виправити таку помилку. Її виправлення можливе при використанні коду з більшою корегуючою здатністю.

Надходження з каналу нового елементарного блоку викликає зрушення картинки у вікні декодування вліво. В результаті ліве ребро шляху зникає, а праворуч з'являється новий стовпець решітки, до вузлів якого повинні бути протягнуті збережені шляхи від вузлів попереднього стовпця. Для цього виконуються такі операції.

Для кожного вузла нового стовпця обчислюються відстані між прийнятим блоком і маркуванням ребер, провідних в даний вузол. Отримані метрики ребер підсумовуються з відстанню шляхів, які вони протягують.

З двох можливих шляхів залишається шлях з меншою метрикою, а другий відкидається, так як наступні поступаючі блоки не можуть змінити співвідношення відстаней цих шляхів. У разі рівності відстаней або випадково вибирається один шлях, або зберігаються обидва.

В результаті цих операцій до кожного вузла нового стовпця знову веде один шлях. Наприклад, нехай новий блок з каналу дорівнює 00. Розглянемо продовження шляху до нижнього вузла решітки, в який можна потрапити зі стану кодера 10 по ребру 01 або зі стану 11 по ребру 10 (див. Рис.3). В обох випадках відстань цих ребер від прийнятого блоку 00 рівна 1. Однак сумарна відстань шляху, продовженого зі стану 10, рівна 6, а шляху зі стану 11 рівна 7. Тому другий шлях буде відкинутий разом з ребром 01, яке входило в нижній вузол на попередньому кроці декодування (див. рис. 4).

Оцінка інформаційного символу проводиться по крайньому лівому ребру шляху в вікні декодування. Згідно з правилом побудови кодової решітки приймається, що інформаційний символ дорівнює 0, якщо це ребро верхнє, і 1, якщо ребро нижнє.

Розглянутий приклад пояснює роботу декодера в припущенні, що вихідний сигнал демодулятора квантується на 2 рівня (так зване жорстке декодування). Більше число рівнів квантування приводить до м'якого декодування. Встановлено, що 8 рівнів квантування гарантують практично потенційну достовірність декодування.

Щоб у цьому випадку використовувати алгоритм Вітербі, потрібно замість відстані Хеммінга ввести нову відстань, яке точніше враховує відмінність між прийнятою послідовністю (вихідним багаторівневим сигналом демодулятора) Y і очікуваним двійковим кодовою словом С. Наприклад, можна використовувати Евклідову відстань або відстань

D:\Personal\Education\Інститут\Magistr\Диплом\Мій\Записка\analit_images\image006.jpg (7.5)

де Р (yi | ci) - перехідна ймовірність каналу, тобто умовна ймовірність появи на виході демодулятора відліку yi при передачі символу ci.

Для зручності обчислень (7.5) на практиці замінюється на

D:\Personal\Education\Інститут\Magistr\Диплом\Мій\Записка\analit_images\image007.jpg

де mi - цілі позитивні числа, кількість яких дорівнює числу рівнів квантування, причому mi = 0, коли між yi і ci спостерігається найбільша відповідність.

На рис. 5 наведено розмічений Граф каналу при 4 рівнях квантування, яким приписані числа mi, рівні 0, 1, 2, 3.

Граф дозволяє визначити міру розбіжності між ребрами шляху на кодової решітці та прийнятими символами. Якщо,

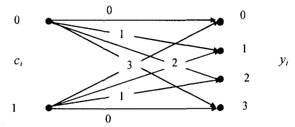


Рис. 5. Граф каналу при м'якому декодуванні

наприклад, на виході демодулятора yi = 2, а маркування ребра перевіряється шляху має ci = 0, то міра розбіжності дорівнює 2. При yi = 2 та наявності на ребрі мітки ci = 1 розбіжність mi = 1. Ці числа сумуються в межах кадру n0 і додаються до міри розбіжності продовжуваного шляху. Подальші операції алгоритму Вітербі при м'якому декодуванні збігаються з операціями жорсткого декодування. Виграш м'якого декодування становить близько 2 дБ. Так як складність обчислень при цьому зростає незначно, то м'яке декодування широко використовується в сучасних системах мобільного зв'язку.